Lappeenrannan teknillinen yliopisto
Sähkötekniikan osasto
Säätö- ja Digitaalitekniikan laboratorio 15.1.2008

KANDIDAATINTYÖ

Reaaliaikaisen tiedonsiirtolinkin muodostaminen prosessisäädön simulointi- ja valvomo-ohjelmiston välille

0259986 Kalle Kukkonen

SISÄLLYSLUETTELO

KÄYTETYT LYHENTEET JA MERKINNÄT 2
1 JOHDANTO 3
2 TIEDONSIIRTOPROTOKOLLA, OHJELMISTOT JA SIMULOITAVA PROSESSI 4
2.1 Dynamic Data Exchange 4
2.2 MATLAB 6
2.3 InTouch 7
2.4 Simuloitava prosessi 9
2.4.1 Prosessin Simulink-malli 12
3 KÄYTTÖLIITTYMÄ JA DDE-LINKKI 17
3.1 Käyttöliittymä 17
3.2 DDE-linkin muodostaminen 18
4 DDE-LINKIN JA KÄYTTÖLIITTYMÄN TOIMINTA 24
5 YHTEENVETO 27
LÄHDELUETTELO 28
LIITTEET I Esimerkkejä DDE-funktioiden käytöstä
II DDE-source lohkon syntaksi
III DDE-sink lohkon syntaksi
IV RT Blockset:in syntaksi

KÄYTETYT LYHENTEET JA MERKINNÄT

COM	Component Object Model
DCOM	Distributed Component Object Model
DDE	Dynamic Data Exchange
HMI	Human-Machine Interface
I/O	input/output
PLC	Programmable Logic Control
OLE	Object Linking and Embedding
OPC	OLE for Process Control
TCP/IP	Transmission Control Protocol / Internet Protocol

A	tankin poikkileikkauksen pinta-ala
a	tankin poistoreiän poikkileikkauksen pinta-ala
g	maan vetovoiman aiheuttama putoamiskiihtyvyys
h	tankin nestepinnankorkeus
p	paine
q	virtaus
v	virtausnopeus
ρ	tiheys

1 JOHDANTO

Teollisuusautomaatiossa käytetään varta vasten automaatiosovelluksiin tarkoitettuja tietokoneita eli ohjelmoitavia logiikoita (PLC, Programmable Logic Control). Ohjelmoitavan logiikan ja käyttäjän välillä on hyvin useasti jonkin asteinen käyttöliittymä (HMI, Human-Machine Interface). Käyttöliittymä voidaan toteuttaa logiikkaan liitettävien näyttöpäätteiden tai PC-pohjaisten valvomo-ohjelmien avulla. Käyttöliittymän kautta käyttäjä voi valvoa ja ohjata automaatiojärjestelmää.

Tämän kandidaatintyön aiheena on reaaliajassa toimivan tiedonsiirtolinkin luominen prosessisäädön simulointiohjelmiston ja valvomo-ohjelmiston välille. Prosessiasäätöä simuloidaan MATLAB:in Simulink-ohjelmistolla ja käyttöliittymä luodaan InTouch valvomo-ohjelmistolla. Simuloitavana prosessina toimii nelitankkiprosessi, jossa kahden tankin pinnankorkeutta säädetään kahdella pumpulla. Prosessista tekee erittäin mielenkiintoisen se, että prosessilla on kolmitieventtiilien asennoista riippuen kaksi eri toimintapistettä: minimivaiheinen ja ei-minimivaiheinen. Myös tankkien ristiinkytköksien johdosta prosessi on normaalia tankkiprosessia mielenkiintoisempi.

Tiedonsiirtolinkin muodostaminen prosessisäädön simuloinnin sekä valvomo-ohjelmiston välille mahdollistaa lukuisia erilaisia käyttötarkoituksia. Varsinkin opetuskäytössä tämä on erittäin käyttökelpoinen, koska se ei vaadi todellisen prosessin eikä laitteistojen läsnäoloa. Sen avulla voidaan opettaa valvomo-ohjelmien luomista sekä niiden käyttöä. Myös prosessisäätöä voidaan opettaa erittäin havainnollisesti.

2 TIEDONSIIRTOPROTOKOLLA, OHJELMISTOT JA SIMULOITAVA PROSESSI

Tiedonsiirtolinkin muodostamiseen on tarjolla useita eri protokollia. Tässä työssä tiedonsiirtoon käytetään DDE-protokollaa. Prosessisäädön simulointi tehdään Simulink:illä ja prosessin käyttöliittymä luodaan InTouch-ohjelmistolla.

2.1 Dynamic Data Exchange

Dynamic Data Exchange (DDE) on Microsoftin suunnittelema kommunikaatioprotokolla, joka mahdollistaa sovelluksien välisen tiedonsiirron Windows-ympäristössä. Se muodostaa kahden samanaikaisesti käytössä olevan sovelluksen välille asiakas-palvelin-yhteyden (client-server). Sovellus, joka toimii palvelimena, tuottaa dataa ja vastaanottaa pyyntöjä muilta sovelluksilta, jotka ovat kiinnostuneet palvelimen tuottamasta datasta. Sovellukset, jotka lähettävät pyyntöjä palvelimelle, toimivat asiakkaina.

DDE:tä käytetään hyvin usein reaaliaikaiseen tiedonsiirtoon. Esimerkkinä voidaan mainita mittaustietojen keruu teollisuusympäristössä. Asiakassovellukset voivat käyttää DDE:tä sekä kertaluontoiseen että jatkuvaan tiedonsiirtoon. Jatkuvan tiedonsiirron tapauksessa päivittynyt tieto siirretään heti kun se on saatavilla. DDE:llä voidaan myös lähettää ohjaustietoja toimilaitteille. Esimerkiksi prosessiautomaatiossa asiakassovellukset voivat lähettää ohjaustietoja säiliösysteemin pumpuille tai venttiileille.

DDE on standardiominaisuus Windows-pohjaisille sovelluksille, jotka tarvitsevat tiedonsiirtolinkkiä toisiin sovelluksiin. DDE-ominaisuuden sisältäviä sovelluksia ovat mm. InTouch ja Microsoft Excel. NetDDE-laajennus mahdollistaa DDE-linkin muodostamisen sovellusten välille, jotka toimivat erillisissä modeemilla tai tietoliikenneverkolla yhdistetyissä tietokoneissa.

Saadakseen dataa toisilta sovelluksilta, tulee asiakassovelluksen muodostaa linkki palvelimena toimivaan sovellukseen. Linkin muodostamiseen tarvitaan sekä palvelinsovelluksen nimi (application name) että käsiteltävän aiheen nimi (topic name).

Kun linkki on muodostettu, voidaan aiheeseen liittyviä tietoja lukea ja kirjoittaa. Asiakassovellus pyytää palvelinsovellusta ilmoittamaan aina kun tietyn tiedon arvo muuttuu. Nämä linkit pysyvät aktiivisina, kunnes joko asiakas tai palvelin lopettaa niiden välisen keskustelun. Tämä on tehokas tapa siirtää tietoa, koska linkin muodostamisen jälkeen tietoa siirtyy sovellusten välillä vain kun tieto päivittyy.[1]

Yksi DDE–ominaisuuden sisältävistä sovelluksista on MATLAB. MATLAB:issa DDE-tiedonsiirtoon käytetään taulukon 1 mukaisia funktioita. Liitteessä I on esitetty lyhyt esimerkki DDE–funktioiden käytöstä.

Taulukko 1. MATLAB:issa käytettävissä olevat DDE–funktiot [2].

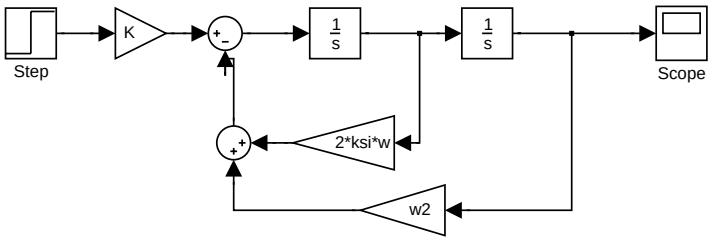
Funktio	**Kuvaus**
ddeadv	Pystyttää aktiivisen linkin MATLAB:in ja palvelinsovelluksen välille.
ddeexec	Lähettää suoritusmerkkijonon palvelinsovellukselle.
ddeinit	Aloittaa MATLAB:in ja toisen sovelluksen välisen DDE-keskustelun.
ddepoke	Lähettää dataa MATLAB:ista palvelinsovellukselle.
ddereq	Pyytää dataa palvelinsovellukselta.
ddeterm	Lopettaa MATLAB:in ja toisen sovelluksen välisen DDE-keskustelun.
ddeunadv	Vapauttaa MATLAB:in ja palvelinsovelluksen välisen aktiivisen linkin.

DDE:n lisäksi on olemassa myös muita samaan käyttötarkoitukseen soveltuvia tiedonsiirtotapoja. Eräs hyvin laajasti käytössä oleva standardi on avoimeen tiedonsiirtoon tarkoitettu OPC (OLE for Process Control). OPC perustuu Microsoftin kehittämille OLE, COM ja DCOM tekniikoille. Standardi määrittelee joukon olioita, liittymiä ja menetelmiä, joita käytetään teollisuuden prosessiautomaatiossa.

2.2 MATLAB

MATLAB (MATrix LABoratory) on The MathWorks:in luoma numeerinen laskentaympäristö ja ohjelmointikieli. MATLAB mahdollistaa matriisien helpon käsittelyn, datan ja funktioiden esittämisen, algoritmien käytön, käyttöliittymien luomisen sekä niiden käyttämisen yhdessä eri ohjelmointikielillä toteutettujen ohjelmien kanssa. MATLAB:iin on saatavilla useita sovelluskohtaisia lisäosia (toolbox), jotka tekevät siitä erittäin laajasti käytetyn ohjelmiston etenkin säätötekniikan alalla.

Simulink on MATLAB:iin integroitu dynaamisten järjestelmien mallintamiseen, simuloimiseen ja analysoimiseen tarkoitettu ohjelmisto. Simulink:illä voidaan analysoida sekä lineaarisia että epälineaarisia järjestelmiä. Järjestelmiä voidaan mallintaa joko jatkuva-aikaisena, diskreettinä tai näiden yhdistelmänä. Mallintaminen tapahtuu graafisen käyttöliittymän avulla, jossa järjestelmä kuvataan lohkokaaviona. Kuvassa 1 on esitetty Simulink:illä tehty erään dynaamisen järjestelmän lohkokaavioesitys, jossa tarkastellaan järjestelmän lähtöä, kun tulona on askelmainen muutos.

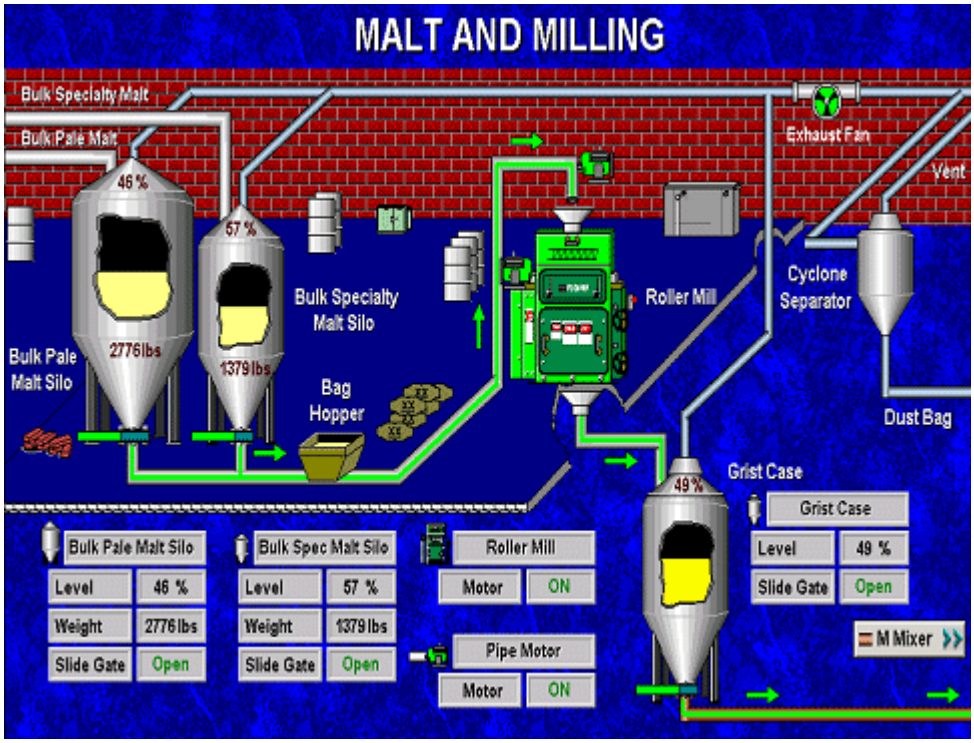


Kuva 1. Erään dynaamisen järjestelmän Simulink–malli.

Simulink tarjoaa laajan valikoiman valmiita lohkoja. Valikoimaan kuuluu mm. signaalilähteitä, lineaarisia ja epälineaarisia komponentteja sekä liitoskappaleita. Myös valmiiden lohkojen muokkaaminen ja omien lohkojen luominen on mahdollista. Simuloinnin tuloksien tarkastelua varten Simulink:in lohkokirjasto sisältää useita erilaisia näyttölohkoja kuten esimerkiksi *Scope*-lohko. Simulointitulokset voidaan siirtää myös MATLAB:iin jatkokäsittelyä ja analysointia varten. Koska MATLAB ja Simulink ovat integroituna toisiinsa, voidaan järjestelmiä simuloida, mallintaa ja analysoida kummassa sovelluksessa tahansa.[3]

2.3 InTouch

InTouch on Wonderware:n valmistama ihmisen ja koneen välisten rajapintojen (HMI) luomiseen tarkoitettu Windows-pohjainen sovellus. InTouch kuuluu Wonderware FactorySuit™ ohjelmistokokonaisuuteen ja on myös yksi tämän alan eniten käytetyistä ohjelmistoista.



Kuva 2. Eräs InTouch:illa luotu HMI-käyttöliittymä [4].

InTouch koostuu kolmesta pääohjelmasta, joita ovat InTouch Application Manager, WindowMaker™ ja WindowViewer™. Application Manager järjestelee käyttäjän luomia sovelluksia. WindowMaker toimii kehitysympäristönä, jossa oliopohjaisella grafiikalla luodaan animoituja näyttöikkunoita. Luotuja ikkunoita voidaan yhdistää teollisiin I/O-järjestelmiin ja muihin Windows-pohjaisiin sovelluksiin. WindowViewer on suoritusympäristö, jossa WindowMaker:illä luotuja näyttöikkunoita voidaan käyttää. WindowViewer suorittaa mm. käyttäjän määrittämiä käskyjä ja loogisia operaatioita, historiatietojen sekä hälytyksien kirjaamista ja raportointia.[5]

InTouch sisältää Script ominaisuuden, jonka avulla käyttäjä voi ohjelmoida luomaansa käyttöliittymään liittyviä komentoja ja loogisia operaatioita. Käyttäjä voi esimerkiksi ohjelmoida ehtolauseita, joiden toteutus riippuu jonkun tietyn painikkeen painamisesta tai halutun kriteerin täyttymisestä luodussa käyttöliittymässä. Ohjelmoinnin avulla voidaan luoda hyvinkin kustomoituja ja automatisoituja käyttöliittymiä.

InTouch:n liittämiseksi automaatiojärjestelmiin on kolme eri mahdollisuutta. Yleisesti käytössä olevista tiedonsiirtotekniikoista InTouch tukee sekä OPC-tekniikkaa että Microsofin DDE–kommunikointia. Näiden kahden lisäksi InTouch voi kommunikoida myös Wonderware:n oman TCP/IP-pohjaisen SuiteLink™ protokollan välityksellä.

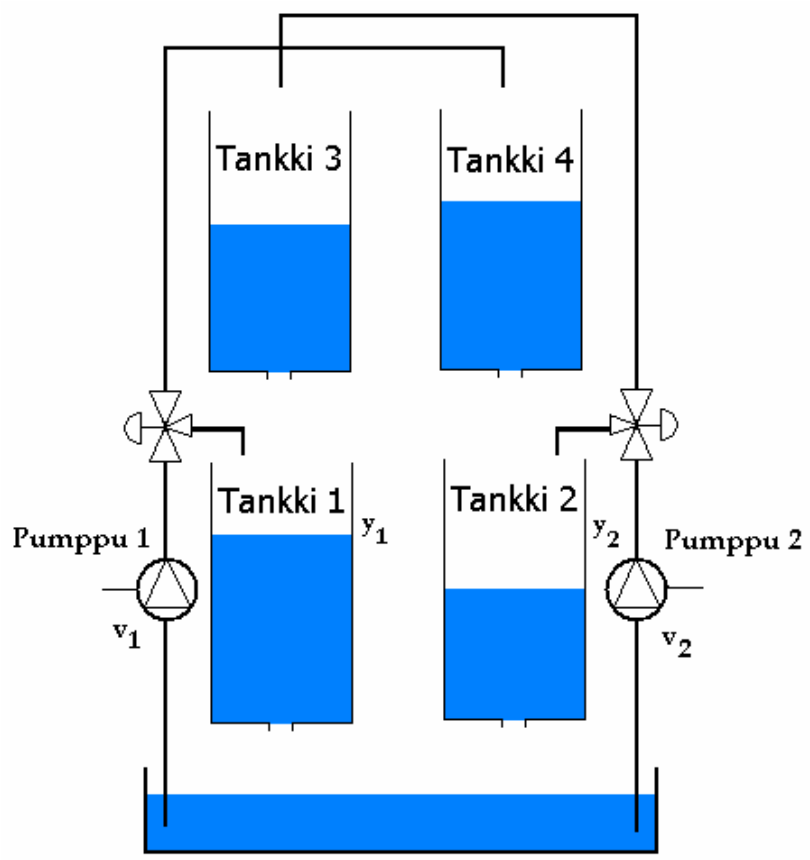
HMI-käyttöliittymien luomiseen tarkoitettuja ohjelmistoja on saatavissa useilta eri valmistajilta. Näistä muutama on esitetty taulukossa 2.

Taulukko 2. Muutamien eri valmistajien HMI-ohjelmistoja.

Valmistaja	**Ohjelmisto**
Siemens	SIMATIC
AutomationDirect	LookoutDirect
SIELCO SISTEMI	Winlog Pro
Open Automation Software	OPC Controls.NET

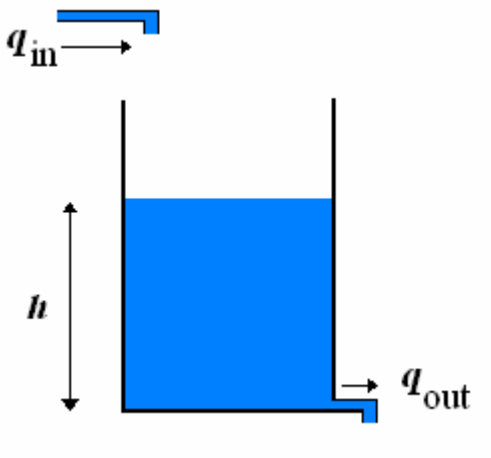
2.4 Simuloitava prosessi

Simuloitavana prosessina on kuvan 3 mukainen neljän tankin prosessi, jossa tarkoituksena on säätää kahden alimman tankin pinnankorkeutta kahdella pumpulla. Ohjaussuureina toimivat pumppujen tulojännitteet (v_1, v_2) ja lähtöinä tankkien pinnankorkeusantureiden antamat jännitteet (y_1, y_2). Prosessia säädetään kahdella PI-säätimellä.



Kuva 3. Nelitankkiprosessin lohkokaavioesitys. Veden pinnan tasoa tankeissa 1 ja 2 ohjataan kahdella pumpulla.

Tarkastellaan kuvan 4 mukaista yhden tankin järjestelmää, jossa tavoitteena on pitää pinnankorkeus h asetusarvossaan. Tankkiin tulevaa virtausta q_{in} asetellaan pumpulla ja tankista lähtevä virtaus q_{out} määräytyy hydrostaattisen paineen perusteella.



Kuva 4. Tankkiprosessi, jossa tavoitteena on pinnankorkeuden h pitäminen asetusarvossaan. Tulovirtaus on q_{in} ja lähtövirtaus q_{out}.

Prosessin dynaaminen malli saadaan Bernoullin yhtälön avulla, jonka mukaan ideaalinesteen paine-energian, liike-energian ja potentiaalienergian summa on vakio [6], eli

$$p + \frac{\rho v^2}{2} + hg\rho = vakio\,, \tag{1}$$

missä p on paine, ρ nesteen tiheys, v virtausnopeus, h pinnankorkeus ja g putoamiskiihtyvyys. Oletetaan, että kyseessä on kokoonpuristumaton neste (ρ = vakio). Tällöin tankkiin virtaavan ja tankista ulosvirtaavan nesteiden energiat ovat yhtä suuret, sillä energiaa ei häviä prosessissa. Tämän perusteella voidaan esittää, että

$$p_1 + \frac{\rho {v_1}^2}{2} + h_1 g\rho = p_2 + \frac{\rho {v_2}^2}{2} + h_2 g\rho\,. \tag{2}$$

Säiliössä virtausnopeus on nolla ja ulosvirtauksessa pinnankorkeus on nolla, mutta paine on sama, joten yhtälö (2) supistuu muotoon

$$\frac{\rho {v_2}^2}{2} = h_1 g\rho\,, \tag{3}$$

josta saadaan lähtovirtauksen nopeudelle yhtälö

$$v_2 = \sqrt{2gh_1} \,. \tag{4}$$

Koska tilavuusvirta on yhtä kuin virtausnopeus kerrottuna poistoreiän poikkileikkauksen pinta-alalla *a*, saadaan lähtövirtaukseksi

$$q_{out} = a\sqrt{2gh} \,. \tag{5}$$

Tankin tilavuustaseelle voidaan esittää yhtälö

$$\frac{dV}{dt} = A\frac{dh}{dt} = q_{in} - q_{out} \,, \tag{6}$$

missä V on tankin tilavuus ja A on tankin poikkipinta-ala. Nyt yhtälöiden (5) ja (6) perusteella saadaan tankin pinnankorkeudelle yhtälö

$$\frac{dh}{dt} = q_{in} - q_{out} = q_{in} - a\sqrt{2gh} \,. \tag{7}$$

Edellä esitetyn perusteella voidaan kuvan 1 mukaisen prosessin tankkien pinnankorkeuksille esittää yhtälöt [7]

$$\begin{aligned}
\frac{dh_1}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}v_1 \\
\frac{dh_2}{dt} &= -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}v_2 \\
\frac{dh_3}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}v_2 \\
\frac{dh_4}{dt} &= -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1
\end{aligned} \,, \tag{8}$$

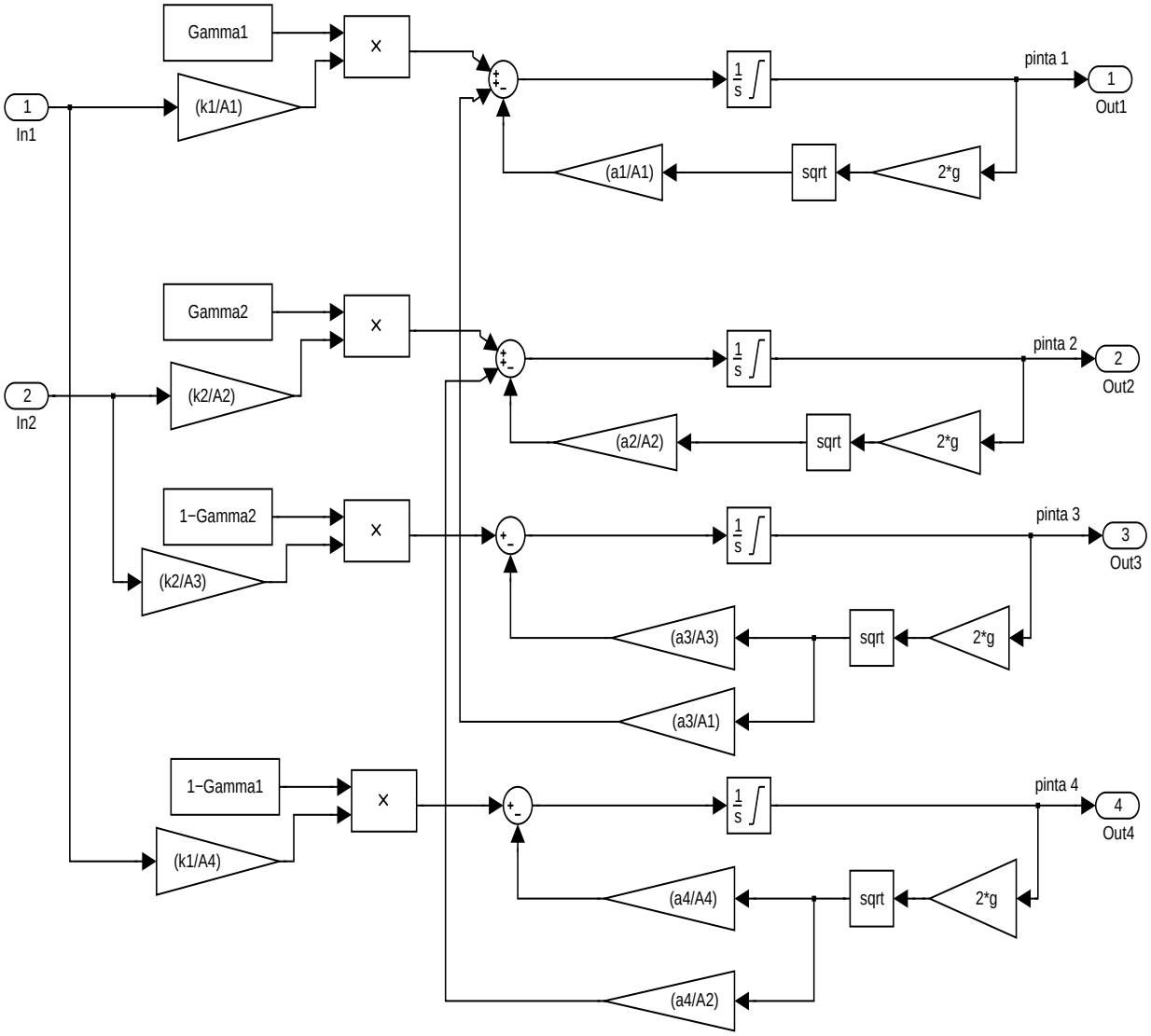
missä pumppuun *i* syötetty jännite on v_i ja siitä aiheutuva virtaus on $k_i v_i$. Parametrit γ_1 ja γ_2 kuvaavat prossessin kolmitieventtiilien asentoa, jotka voivat saada arvoja väliltä 0…1. Virtaus tankkiin 1 on $\gamma_1 k_1 v_1$ ja tankkiin 4 $(1-\gamma_1)k_1 v_1$. Virtaukset tankkeihin 2 ja 3 ovat vastaavasti $\gamma_2 k_2 v_2$ ja $(1-\gamma_2)k_2 v_2$. Parametrit k_1 ja k_2 kuvaavat pumppujen tuottamia virtauksia ja riippuvat ainostaan käytössä olevista pumpuista. Vedenpinnan korkeutta mittaavat signaalit ovat $k_c h_1$ ja $k_c h_2$, missä k_c kuvaa lähtöjännitteen ja pinnankorkeuden välistä suhdetta [7]. Prosessia kuvaavien parametrien arvot on esitetty taulukossa 3.

Taulukko 3. Prossessia kuvaavien parametrien arvot [7].

A_1, A_3	28 [cm^2]
A_2, A_4	32 [cm^2]
a_1, a_3	0,071 [cm^2]
a_2, a_4	0,057 [cm^2]
g	981 [cm/s^2]
k_1, k_2	1 [cm^3/Vs]
k_c	1 [V/cm]

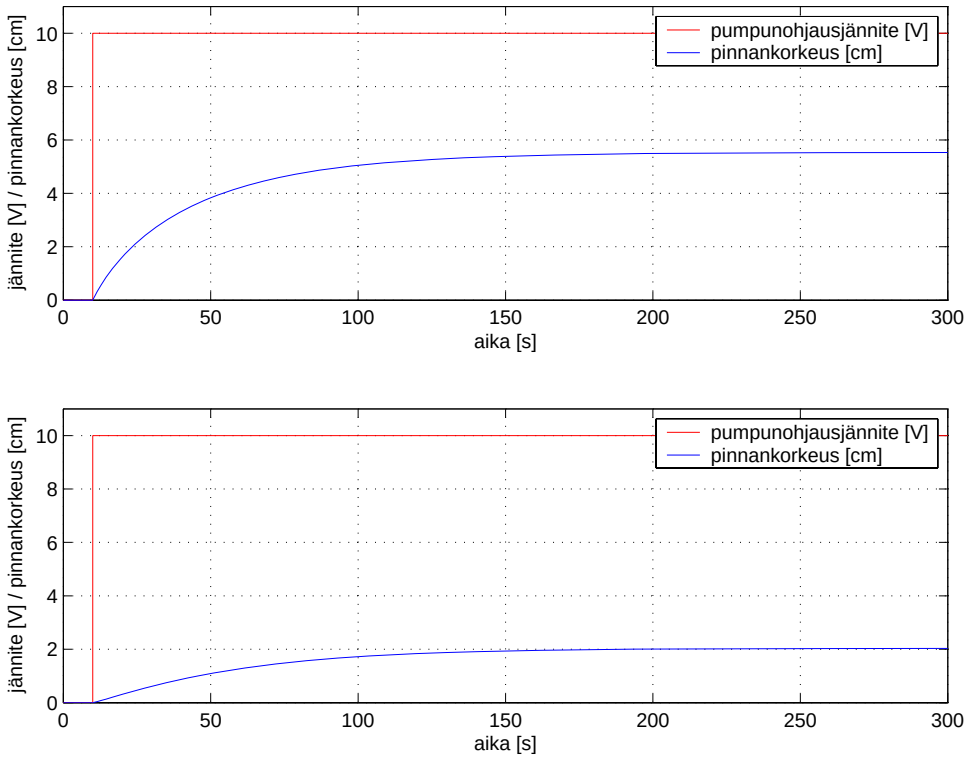
2.4.1 Prosessin Simulink-malli

Nelitankkiprosessille esitettyjen yhtälöiden (8) perusteella voidaan muodostaa prosessille Simulink-malli, jossa tulosignaaleina ovat pumpujen ohjausjännitteet ja lähtösignaaleina ovat tankkien 1…4 pinnankorkeudet. Prosessin Simulink-malli on esitetty kuvassa 5. Parametrit *Gamma1* (γ_1) ja *Gamma2* (γ_2) kuvaavat prosessin kolmitieventtiilien asentoa.



Kuva 5. Nelitankkiprosessin Simulink-malli. Tuloina ovat pumppujen 1 ja 2 ohjausjännitteet ja lähtöinä tankkien 1…4 pinnankorkeudet.

Simuloimalla edellä esitettyä mallia venttiilien asennoilla *Gamma 1* = 0,7 ja *Gamma 2* = 0,6 saadaan tankkien 1 ja 2 pinnankorkeuksille kuvassa 6 esitetyt askelvasteet. Kuvan 6 perusteella voidaan todeta, että tehtäessä pumppujen 1 ja 2 ohjausjännitteeseen askelmainen muutos 0 V…10 V, nousee tankin 1 pinnankorkeus 5,5 cm:ä ja tankin 2 2 cm:ä. Pinnankorkeuksien keskinäinen ero johtuu ainoastaan simuloinnissa käytetyistä venttiilien asennoista.

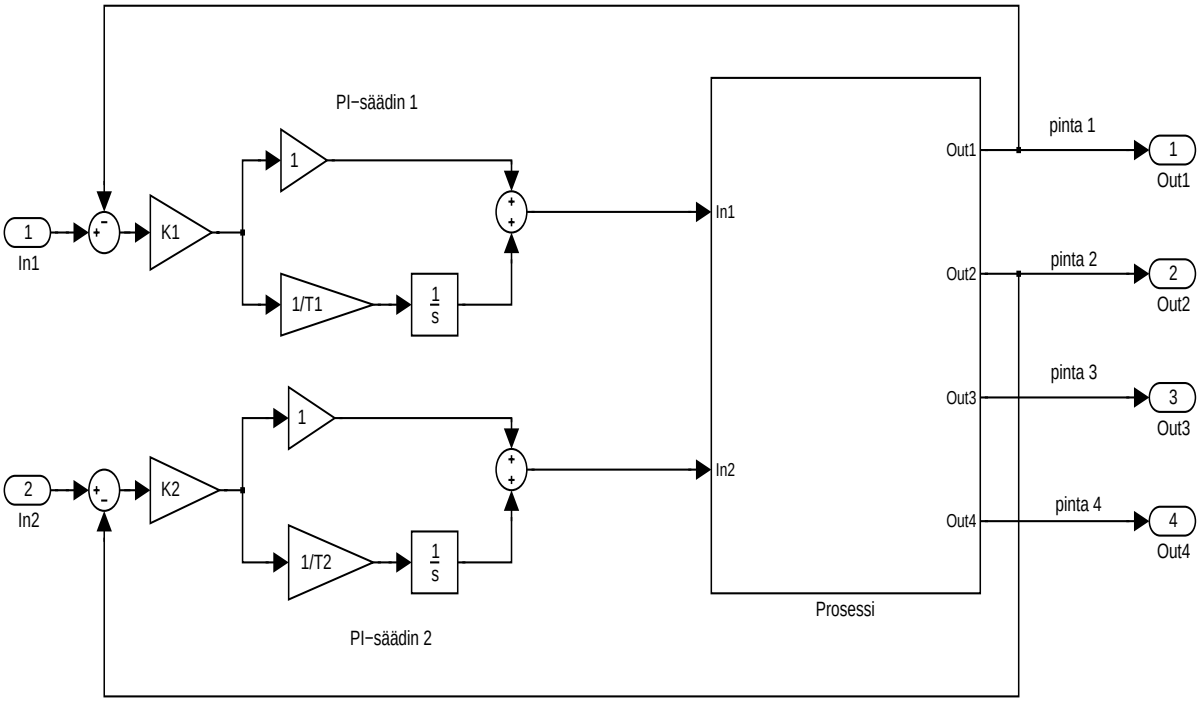


Kuva 6. Tankkien 1 (ylempi) ja 2 (alempi) pinnankorkeuksien vasteet, kun molempien pumppujen ohjausjännitettä muutetaan askelmaisesti 0 V…10 Vajanhetkellä 10s.

Pinnankorkeussäätöjen toteuttamiseen käytetään PI-säätöä. Koska tarkoituksena on säätää kahta eri tankkia, on myös säätimiä ja takaisinkytkentöjä kaksi. Säädön simulointia varten muodostetaan kuvan 7 mukainen Simulink-malli. Säätimien tuloina ovat operaattorilta saatavat pinnankorkeuksien ohjearvot. Prosessina toimii edellä muodostettu simulointimalli, joka on esitetty kuvassa 5. Säätimien parametreina käytettiin taulukon 4 mukaisia arvoja.

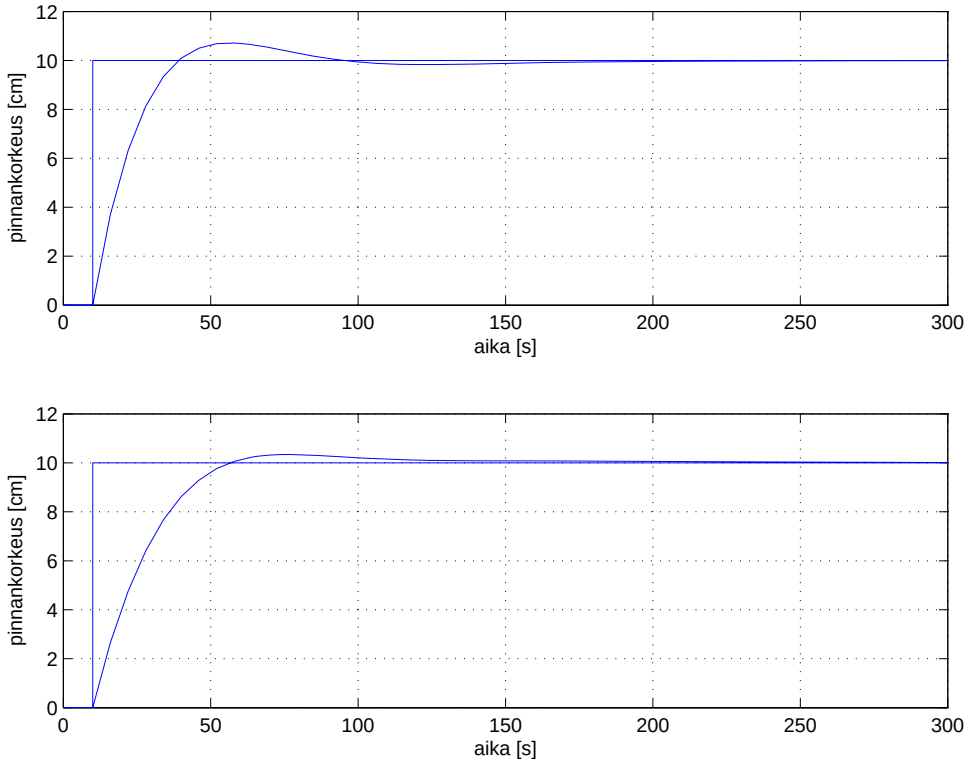
Taulukko 4. PI-säätimien parametrit, kun venttiilien asennot ovat $\gamma_1 = 0{,}7$ ja $\gamma_2 = 0{,}6$ [7].

(K_1, K_2)	(3,00, 2,70)
(T_1, T_2)	(30, 40)



Kuva 7. Prosessille muodostettu takaisinkytketty säätörakenne kahdella PI-säätimellä, jossa takaisinkytkennät on otettu tankkien 1 ja 2 pinnankorkeuksista.

Lisäämällä prosessiin pinnankorkeussäädöt saadaan pinnankorkeuksille kuvassa 8 esitetyt vasteet. Kuvasta 8 voidaan huomata, että nyt molempien tankkien pinnankorkeudet asettuvat 10 cm:iin, kun niille annetaan askelmainen muutos 0 cm…10 cm.



Kuva 8. Säädetyn prosessin tankkien pinnankorkeuksien vasteet. Tankki 1 (ylempi) ja tankki 2 (alempi).

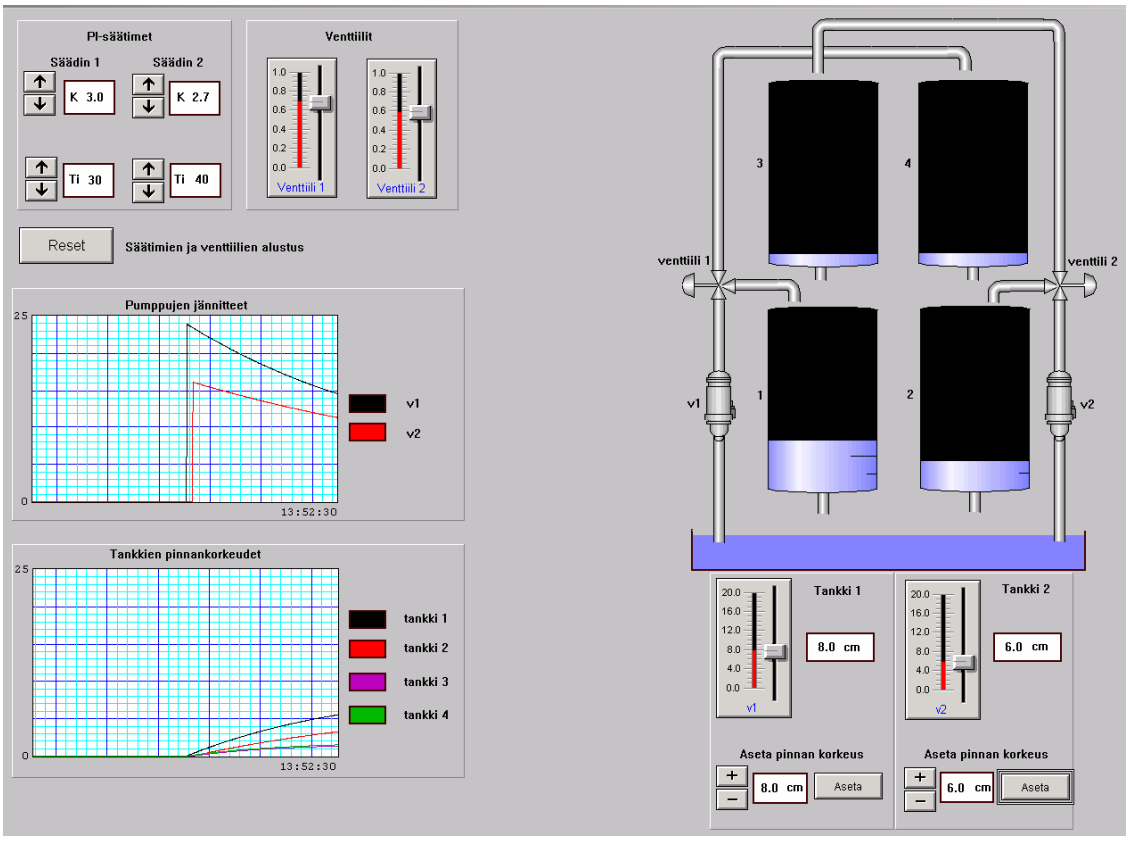
3 KÄYTTÖLIITTYMÄ JA DDE-LINKKI

Prosessisäädölle luodaan graafinen käyttöliittymä InTouch-ohjelmistolla. Luotu käyttöliittymä yhdistetään edellä esitettyyn simulointimalliin DDE-linkin välityksellä.

3.1 Käyttöliittymä

Edellä esitetyn prosessin ohjaamista ja monitorointia varten luodaan prosessille käyttöliittymä InTouch WindowMaker-sovelluksella. Käyttöliittymästä on pystyttävä antamaan tankkien 1 ja 2 pinnankorkeuksille ohjearvoja sekä tarkkailemaan tankkien pinnankorkeuksia. Myös säätimien ja venttiilien parametrien arvot tulee olla muutettavissa käyttöliittymästä käsin.

Prosessille luotu käyttöliittymä on esitetty kuvassa 9. Käyttöliittymässä tankkien pinnankorkeuksia voidaan säätää portaattomasti liukukytkimillä sekä *aseta*-napeilla. Molempien säätimien parametrien arvoja voidaan muuttaa nuolipainikkeilla. Venttiilien sulkemista ja avaamista voidaan säätää liukukytkimillä. Säätimien ja venttiilien alustaminen taulukossa 4 esitettyihin arvoihin tapahtuu *Reset*-painikkeella. Tankkien pinnankorkeuksien ja pumppujen jännitteiden muutoksien seuraamista varten on käyttöliittymässä kaksi kuvaajaa, jotka päivittyvät prosessin rinnalla. Käyttöliittymän tankkien maksimi pinnankorkeus on 20 cm.



Kuva 9. Prosessin ohjaamista ja monitorointia varten luotu käyttöliittymä.

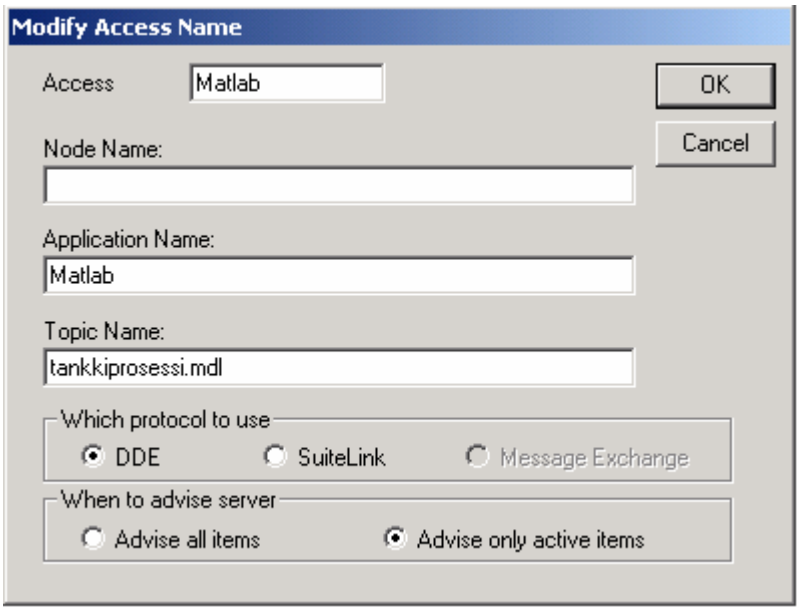
3.2 DDE-linkin muodostaminen

Käyttöliittymän ja Simulink-mallin keskenäinen kommunikointi toteutetaan DDE-linkin välityksellä. Linkin lähtökohdaksi otetaan asetelma, jossa InTouch toimii palvelimena ja MATLAB asiakkaana. DDE-linkkiä varten määritellään sovellusten väliset signaalit sekä näiden kulkusuunnat. Linkin muodostamiseen tarvittavat signaalit on esitetty taulukossa 5.

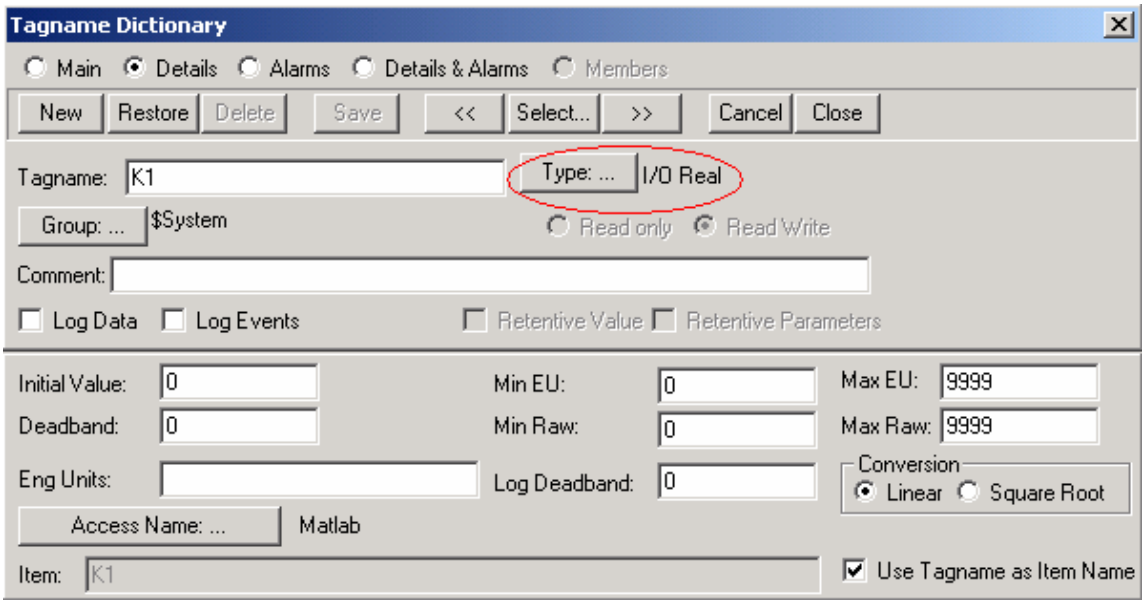
Taulukko 5. InTouch:in ja MATLAB:in väliset signaalit sekä niiden kulkusuunnat.

Prosessimuuttuja	**Signaali**	**Kulkusuunta**
Tankin 1 pinnankorkeuden ohjearvo	yref1	InTouch→MATLAB
Tankin 2 pinnankorkeuden ohjearvo	yref2	InTouch→MATLAB
Venttiilin 1 asento (0...1)	valve1	InTouch→MATLAB
Venttiilin 2 asento (0...1)	valve2	InTouch→MATLAB
Säätimen 1 vahvistus	K1	InTouch→MATLAB
Säätimen 1 integrointiaika (1/T1)	Ti1	InTouch→MATLAB
Säätimen 2 vahvistus	K2	InTouch→MATLAB
Säätimen 2 integrointiaika (1/T2)	Ti2	InTouch→MATLAB
Pumpun 1 jännite	pump1	MATLAB→InTouch
Pumpun 2 jännite	pump2	MATLAB→InTouch
Tankin 1 pinnankorkeus	level1	MATLAB→InTouch
Tankin 2 pinnankorkeus	level2	MATLAB→InTouch
Tankin 3 pinnankorkeus	level3	MATLAB→InTouch
Tankin 4 pinnankorkeus	level4	MATLAB→InTouch

DDE-tiedonsiirto InTouch:in ja MATLAB:in välillä onnistuu hyvinkin yksinkertaisesti taulukossa 1 esitettyjen MATLAB funktioiden avulla. InTouch:issa linkin muodostamista varten on signaalien *access name* määriteltävä kuvan 10 mukaisesti. Signaaleille määritellään sekä sovelluksen että aiheen nimi, johon tiedonsiirto liittyy. Signaalien tyypit on myös määriteltävä I/O-tyyppisiksi, kuten on esitetty kuvassa 11.



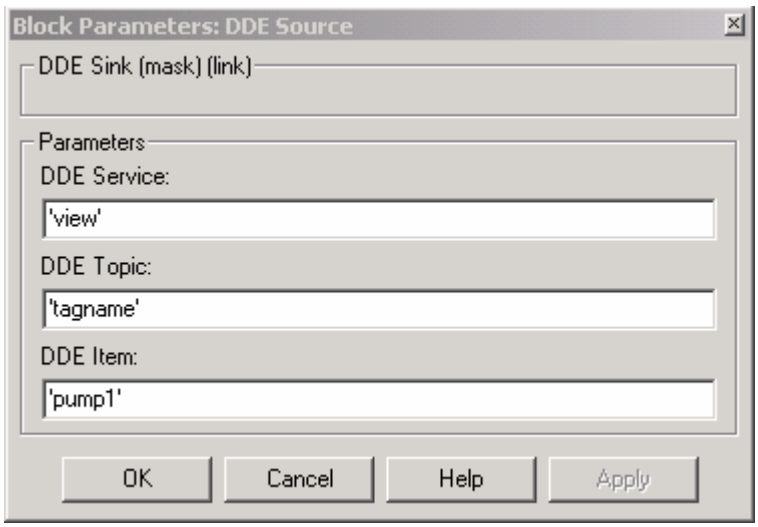
Kuva 10. DDE-linkin muodostamiseen tarvittavat määritykset InTouch:issa.



Kuva 11. Signaalin tyypin määrittäminen.

Koska prosessia simuloidaan Simulink:issä, on tiedonsiirto Simulink:in ja InTouch:in välillä oleellisempaa kuin tiedonsiirto MATLAB:in ja InTouch:in välillä. Tämä ei kuitenkaan onnistu pelkkien DDE-funktioiden avulla. Suurimpana ongelmana on tiedonsiirto Simulink-mallin kanssa simuloinnin ollessa käynnissä. Tähän tarkoitukseen ei sovellu mikään Simulink:in valmiista lohkoista. Ratkaisu ongelmaan löytyy MathWork:sin ylläpitämältä MATLAB Central internetsivustolta [8]. Sivustolla on ladattavissa valmis DDE-Simulink-kirjasto [9], joka sisältää *DDE-sink* ja *DDE-source*

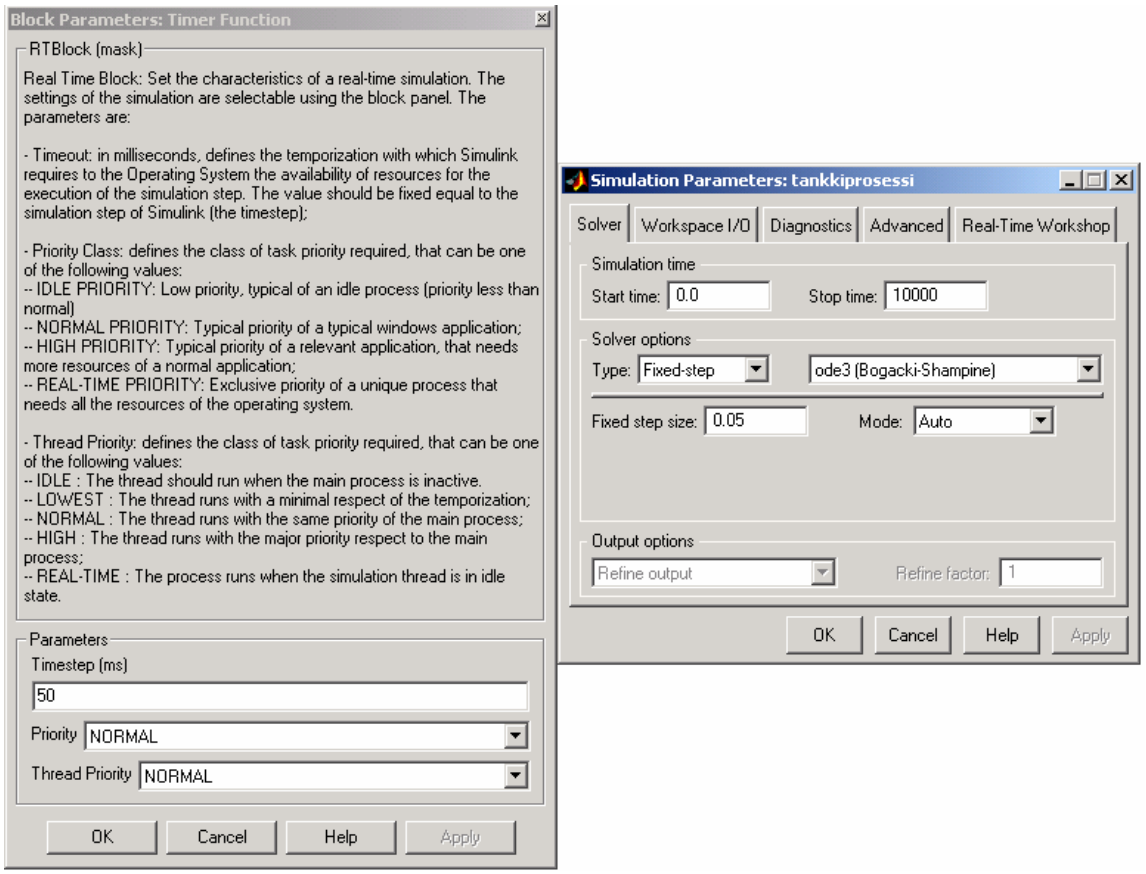
lohkot. Molemmat lohkot ovat m-tiedostona koodattuja s-funktiota ja ne on toteutettu taulukossa 1 esitettyjen funktioiden avulla. *DDE-source* lohkon syntaksi on esitetty liitteessä II ja *DDE-sink* lohkon liitteessä III. DDE-linkin muodostamiseksi InTouch:in kanssa on sekä *source*- että *sink*-lohkoille määriteltävä yhteysparametrit kuvan 11 mukaisesti. Määriteltäviä parametreja ovat sovelluksen nimi, tiedonsiirron aihe sekä signaalin nimi.



Kuva 11. *DDE-source* ja –sink lohkoille määritettävät parametrit.

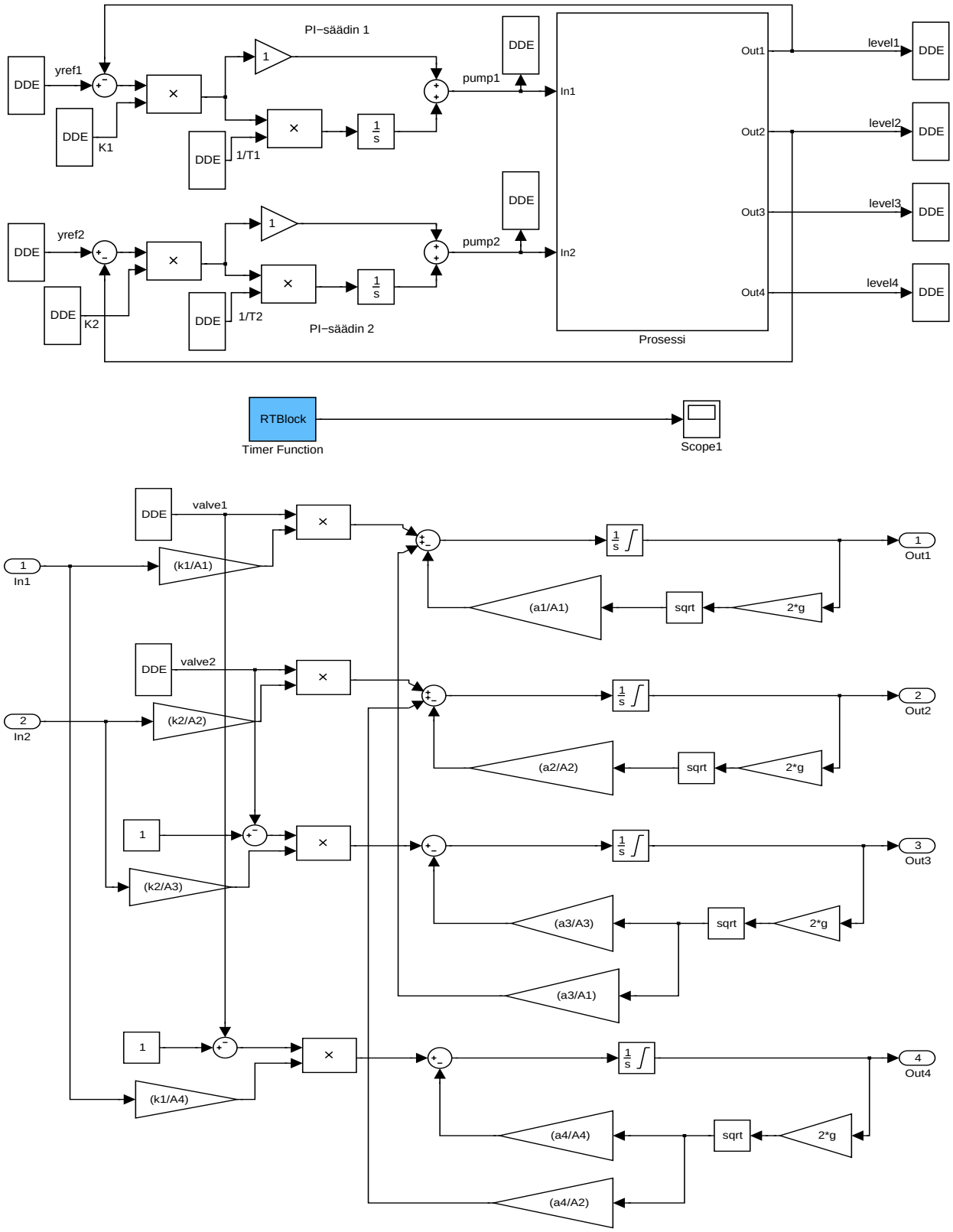
Toinen ongelma linkin muodostamisessa on se, että miten Simulink-malli saadaan pyörimään InTouch:in rinnalla. Yksinkertaisin keino tämän toteuttamiseksi on asettaa simuloinnin päättymisajaksi *Inf* eli ääretön. Tällöin Simulink simuloi prosessia päättymättömästi ja sitä voidaan simuloida käyttöliittymän pyöriessä rinnalla samanaikaisesti. Jotta simulointimalli vastaisi mahdollisimman hyvin alkuperäistä prosessia, on simulointi saatava pyörimään reaaliajassa. Yksinkertainen tapa reaaliaikaisuuden toteuttamiseksi on ladata MATLAB Central:in internetsivustoilta [8] löytyvä RT Blockset [10]. RT Blockset on toteutettu C++ ohjelmointikielellä koodattuna s-funktiona, jonka syntaksi on esitetty liitteessä IV. Blockset:in toiminta perustuu siihen, että se viivästyttää simulaation yhtä askelta niin kauan, että simulointiaskeleen pituudeksi määritetty aika on kulunut. Jotta tämä onnistuu, on sekä Blockset:in sisältämän *Timer function*-lohkolle että Simulink-mallille määritettävä samanpituiset aika-askeleet kuvan 12 mukaisesti. Nyt simulaation kesto reaaliajassa voidaan määrittää

yksinkertaisesti muuttamalla simulaation alkamis- ja päättymisaikaa. Kuvan 12 tapauksessa simulointi kestää 10 000 s.



Kuva 12. Aika-askeleiden määrittäminen *Timer function*-lohkolle (vas.) ja Simulink-mallille (oik.).

Lisäämällä kuvissa 5 ja 7 esitettyihin Simulink-malleihin sekä *DDE*- että *Timer function*-lohkot saadaan aikaan malli, joka pyörii reaaliajassa ja lähettää sekä vastaanottaa dataa DDE-linkin välityksellä. Tämä malli on esitetty kuvassa 13.

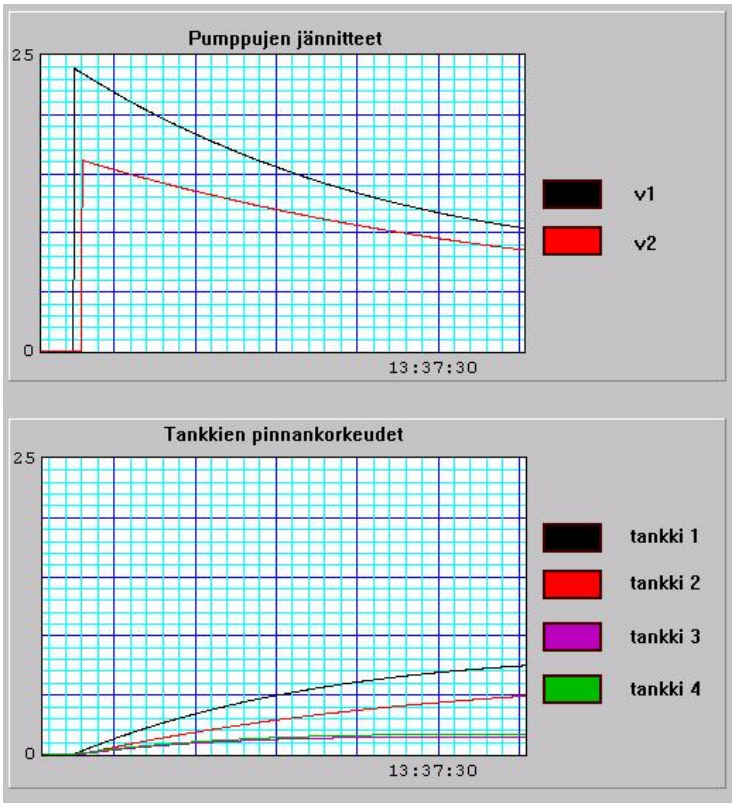


Kuva 13. Säätörakenteen (ylempi) ja prosessin (alempi) Simulink-mallit. Mallit pyörivät reaaliajassa ja lähettävät sekä vastaanottavat dataa DDE-linkin välityksellä.

4 DDE-LINKIN JA KÄYTTÖLIITTYMÄN TOIMINTA

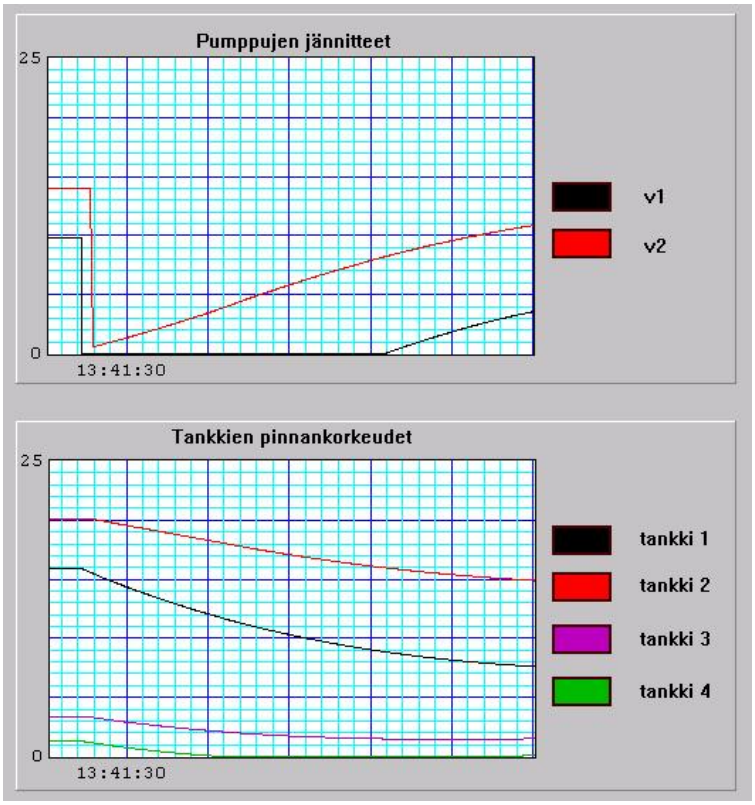
DDE-linkin muodostamiseksi tulee käyttöliittymä ottaa käyttöön avaamalla se WindowViewer-ohjelmalla. Tämän jälkeen käynnistetään Simulink:istä luotu simulaatiomalli. Nyt sekä käyttöliittymä että simulaatiomalli pyörivät ja niiden välinen DDE-kommunikointi on muodostettu.

Tiedonsiirtolinkin toimintaa voidaan demonstroida muuttamalla ohjearvoja sekä parametreja käyttöliittymästä käsin. Asettamalla tankin 1 pinnankorkeudeksi 8 cm:ä ja tankin 2 pinnakorkeudeksi 6 cm:ä, voidaan kuvasta 14 nähdä, että DDE-linkin välityksellä ohjearvot siirtyvät simulointimalliin, joka puolestaan siirtää tankkien pinnankorkeuksien sekä pumppujen jännitteiden muutokset takaisin käyttöliittymään. Simulointimalli alkaa välittömästi reagoida annettuihin ohjearvoihin. Säätimen ja venttiilien parametrit ovat taulukon 4 mukaisia.



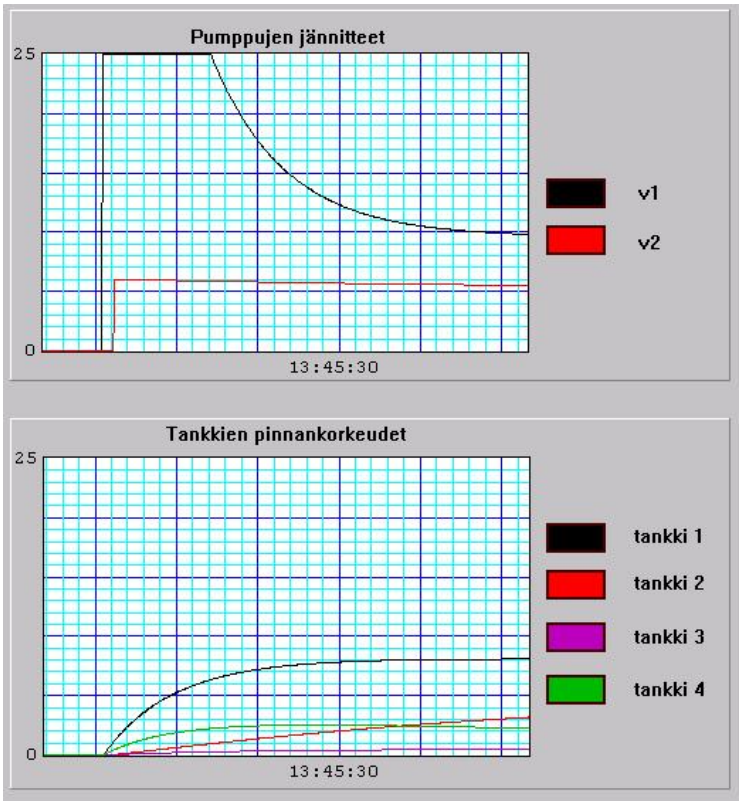
Kuva 14. Simulointimallista välittyneet pinnankorkeuksien muutokset käyttöliittymästä annettujen ohjearvojen perusteella.

Kuvassa 15 on esitetty edellistä vastaava tilanne, mutta nyt tankkien ohjearvot muuttuvat 16 cm→8 cm ja 20 cm→ 15 cm.



Kuva 15. Tankkien pinnankorkeuksien muutos annettujen ohjearvojen perusteella.

Myös prosessin parametrien (säätimet, venttiilit) muuttaminen onnistuu linkin välityksellä. Kuvassa 16 on esitetty tilanne, jossa säätimen 1 vahvistusta on muutettu 3,0→10,0 ja säätimen 2 vahvistusta 2,7→1. Nyt huomataan, että verrattaessa tilannetta kuvan 14 tilanteeseen, reagoi tankin 1 pinnankorkeus ohjearvon muutokseen paljon nopeammin, kun taas tankin 2 pinnankorkeuden muutos on todella hidasta. Muutoksen vaikutus on nähtävissä myös pumppujen jännitteissä.



Kuva 16. Tankkien pinnankorkeuksien reagointi ohjearvoon, kun säätimen 1 vahvistusta on muutettu 3,0→10,0 ja säätimen 2 vahvistusta 2,7→1

5 YHTEENVETO

Reaaliaikaisen tiedonsiirtolinkin muodostaminen prosessisäädön ja valvomo-ohjelmiston välille onnistuu DDE-protokollan avulla helposti. DDE on erittäin tehokas ja hyvin yksinkertainen tapa siirtää tietoa Windows-ympäristössä. Reaaliajassa toimivan DDE-linkin muodostaminen Simulink:in ja InTouch:in välillä onnistuu hyvinkin vaivattomasti MATLAB Central-sivustolta löytyvien DDE-kirjaston sekä RT Blockset:in avulla. InTouch on hyvä ohjelmisto valvomo-ikkunoiden luomiseen, jolla graafisten ja animoitujen käyttöliittymien luominen onnistuu jo pienen opettelun jälkeen. InTouch sisältää laajan valikoiman valmiita symboleja, piirrosmerkkejä, painikkeita sekä mittareita, joiden avulla monimutkaisten ja näyttävien käyttöliittymien luominen onnistuu kohtuullisen helposti.

Reaaliaikaisen tiedonsiirtolinkin muodostaminen prosessisäädön simulaation ja valvomo-ohjelmiston välillä mahdollistaa useita erilaisia sovelluksia. Selvimpiä käyttökohteita ovat automaatioon liittyvät opetuskäytöt. Linkin muodostaminen mahdollistaa havainnollisen tavan opettaa sekä valvomo-ohjelmien luomista että luotujen valvomo-ohjelmistojen käyttöä ilman tarvetta todellisen prosessin tai laitteistojen läsnäololle. Myös prosessisäädön suunnittelun opettamiseen on nähtävissä suurta hyötyä tiedonsiirtolinkin muodostamisesta.

LÄHDELUETTELO

[1] Wonderware FactorySuite. NetDDE user guide.

[2] The MathWorks Inc. Application Program Interface Guide. [viitattu 10.11.2007]. Saatavissa: www.unc.edu/depts/case/BMELIB/apiguide.pdf

[3] Center for Coastal Studies. MATLAB Help Desk. Using Simulink. [viitattu 8.10.2007]. Saatavissa: http://www-ccs.ucsd.edu/matlab/pdf_doc/simulink/sl_using.pdf

[4] PLC Automation Inc. [viitattu 26.11.2007]. Saatavissa: http://www.plcintegrator.com/images/Malt_millbg.gif

[5] Wonderware FactorySuite. InTouch users guide.

[6] Suomen Säätöteknillinen Seura ry, Paine ja Virtaus. INSKO ja Insinööritieto Oy, 1981. s.108-111. ISBN 951-793-383-5.

[7] K. H. Johansson. The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero. IEEE Transaction on Control System Technology, vol 8, no. 3, 3.3.2000.

[8] The MathWorks Inc. MATLAB Central. [viitattu 26.11.2007]. Saatavissa: http://www.mathworks.com/matlabcentral

[9] DDE-Simulink-kirjasto. MATLAB Central. [viitattu 28.11.2007] Saatavissa: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3376&objectType=file

[10] RT Blockset. MATLAB Central. [viitattu 28.11.2007] Saatavissa: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3175&objectType=file

MATLAB:in ja Excelin välisen DDE–keskustelun aloittaminen tapahtuu funktiolla *ddeinit*. Funktio annetaan parametreina ohjelman nimi (excel.exe) sekä käsiteltävän tiedoston nimi (taulukko1.xls).

```
channel = ddeinit('excel', 'taulukko1.xls')
```

Datan pyytäminen Excel–taulukosta suoritetaan funktiolla *ddereq*, joka yksinkertaisimmillaan saa parametrit edellä luotuun keskusteluun sekä halutun datan sijaintiin. Tässä tapauksessa haluttu data sijaitsee taulukon kolmannella rivillä (r3)ja neljännessä sarakkeessa (c4). Siirretty data tallentuu muuttujaan *data*.

```
data = ddereq(channel, 'r4c4')
```

Vastaavasti datan lähettäminen Excel–taulukkoon tapahtuu funktiolla *ddepoke*, mutta edellisestä kohdasta poiketen tulee funktiolle antaa kolmas parametri, joka sisältää lähetettävän datan. Nyt lähetettävä data on muuttujassa *luku* ja sen halutaan sijoittaa taulukon viidennelle riville ja seitsemännelle sarakkeelle. Jos lähetys onnistuu, palauttaa funktio arvon 1 muuttujaan *rc*. Lähetyksen epäonnistuessa palauttaa funktio arvon 0.

```
rc = ddepoke(channel, 'r5c7', luku)
```

```
function [sys,x0,str,ts] = sfunddi(t,x,u,flag,service,topic,item)
%SFUNDDI Simulink DDE source.

switch flag

  case 0;
   [sys,x0,str,ts]=mdlInitializeSizes(service,topic);

  case 2;
    sys = mdlUpdate(t,x,u);

  case 3;
    sys = mdlOutputs(t,x,u,item);

  case 9;
    sys = mdlTerminate(t,x,u);

  otherwise;
    sys=[];

end

function [sys,x0,str,ts]=mdlInitializeSizes(service,topic)

sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

str = [];
ts = [-1 0]; % Inherited sample time

x0 = ddeinit(service,topic);
if (x0==0)
  error('DDE initialization failed.');
end;

function sys = mdlUpdate(t,x,u)

sys = x;

function sys = mdlOutputs(t,x,u,item)

sys = ddereq(x, item);

function sys = mdlTerminate(t,x,u)

ddeterm(x);
sys = [];
```

```
function [sys,x0,str,ts] = sfundde(t,x,u,flag,service,topic,item)
%SFUNDDE Simulink DDE sink.

switch flag

  case 0;
   [sys,x0,str,ts]=mdlInitializeSizes(service,topic);

  case 2;
    sys = mdlUpdate(t,x,u,item);

  case 9;
    sys = mdlTerminate(t,x,u);

  otherwise;
    sys=[];

end

function [sys,x0,str,ts]=mdlInitializeSizes(service,topic)

sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = 0;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);

str = [];
ts = [-1 0]; % Inherited sample time

x0 = ddeinit(service,topic);
if (x0==0)
  error('DDE initialization failed.');
end;

function sys = mdlUpdate(t,x,u,item)

ddepoke(x, item, u);
sys = x;

function sys = mdlTerminate(t,x,u)

ddeterm(x);
sys = [];
```

```
/*
 * File : RTBlock.c
 */

#define S_FUNCTION_NAME  RTBlock
#define S_FUNCTION_LEVEL 2

#include <stdio.h>
#include "simstruc.h"

#define NUMBER_OF_ARGS  3
#define STEP            (int)(*mxGetPr(ssGetSFcnParam(S,0)))
#define PRIORITY        (int)(*mxGetPr(ssGetSFcnParam(S,1)))
#define THPRIORITY      (int)(*mxGetPr(ssGetSFcnParam(S,2)))

// Lib functions definition
void mdlInitializeSizes_Fun(SimStruct *S);
void mdlStart_Fun(SimStruct *S, int priority, int thpriority);
void mdlUpdate_Fun(SimStruct *S, int_T tid, int step, double time);
void mdlTerminate_Fun(SimStruct *S);

static char_T msg[256];

/*====================================================================
=======
+FUNCTION: mdlInitializeSizes

+DESCRIPTION: Setup sizes of the various vectors.

+PARAMETERS:
SimStruct *S

+RETURN: static void
====================================================================
========/
void mdlInitializeSizes(SimStruct *S)
{

#ifdef MATLAB_MEX_FILE
      ssSetNumSFcnParams(S, NUMBER_OF_ARGS);
```

```
if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
      {
            sprintf(msg, "Error in RTBlock: \n");
            sprintf(&msg[strlen(msg)],
                        "Wrong number of input arguments passed.\n%d
arguments are expected\n",
                        NUMBER_OF_ARGS);
            ssSetErrorStatus(S,msg);
            return;
      }
#endif

      if (!ssSetNumOutputPorts(S, 1)) return;
      if (!ssSetNumInputPorts(S, 0)) return;
      ssSetOutputPortWidth(S, 0, 2); //Width of output port one (index
0)

      mdlInitializeSizes_Fun(S);
}



/*********************************************************************
*******
+FUNCTION: mdlInitializeSampleTimes

+DESCRIPTION: Specifiy that we inherit our sample time from the driving
block.

+PARAMETERS:
SimStruct *S

+RETURN: static void
*********************************************************************
********/
void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/*********************************************************************
*******
+FUNCTION: mdlStart

+DESCRIPTION: Routine used to initialize data

+PARAMETERS:
SimStruct *S

+RETURN: static void
*********************************************************************
********/
#define MDL_START  /* Change to #undef to remove function */
void mdlStart(SimStruct *S)
{
      int priority = PRIORITY; // In milliseconds
```

```
      int thpriority = THPRIORITY; // In milliseconds
      mdlStart_Fun(S, priority, thpriority);
}

/*********************************************************************
*******
+FUNCTION: mdlOutputs

+DESCRIPTION:

+PARAMETERS:
SimStruct *S
int_T tid

+RETURN: static void
*********************************************************************
********/
void mdlOutputs(SimStruct *S, int_T tid)
{
}

/*********************************************************************
*******
+FUNCTION: mdlOutputs

+DESCRIPTION:

+PARAMETERS:
SimStruct *S
int_T tid

+RETURN: static void
*********************************************************************
********/
#define MDL_UPDATE
void mdlUpdate(SimStruct *S, int_T tid)
{
      int step = STEP; // In milliseconds
      double time = ssGetT(S);
      mdlUpdate_Fun(S, tid, step, time);
}

/*********************************************************************
*******
+FUNCTION: mdlTerminate

+DESCRIPTION: No termination needed, but we are required to have this
routine.

+PARAMETERS:
SimStruct *S

+RETURN: static void
*********************************************************************
********/
void mdlTerminate(SimStruct *S)
{
```

```
      mdlTerminate_Fun(S);
}

#ifdef  MATLAB_MEX_FILE     /* Is this file being compiled as a MEX-
file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```